Lucrarea de laborator nr. 7

Serializarea

Termenul de **serializare** se foloseşte în limbajele orientate pe obiecte în legătură cu persistenţa obiectelor (instanţelor). Pentru ca un obiect să fie persistent, el trebuie salvat pe o memorie nevolatilă (de exemplu, pe hard disk, în general sub formă de fişier) pentru a putea fi restaurat ulterior pe baza informaţiilor salvate.

Datele instanţei serializate sunt salvate în ordine, de obicei cîmp cu cîmp; dacă se doreşte serializarea mai multor instanţe, salvarea datelor acestora se realizează în ordine, una după alta. Serializarea se face secvenţial, deci ordinea de salvare trebuie să fie aceeaşi cu ordinea de citire pentru restaurare. Nu se pot accesa înregistrări individuale, deci serializarea nu se poate folosi pentru stocarea bazelor de date.

Deserializarea reprezintă operaţia inversă serializării prin care datele serializate în prealabil sunt inserate într-o instanţă (sau în instanţe) ale clasei.

În procesul de serializare/deserializare se pot utiliza diferite standarde, dintre care cele mai cunoscute standarde de industrie sunt fişierele bitmap şi fişierele XML (Extensible Markup Language). NET Framework pune la dispozitie clase specifice pentru operaţiile de serializare/deserializare: `BinaryFormatter` pentru serializarea binară (spațiul de nume `System.Runtime.Serialization.Formatters.Binary`) respectiv `XMLSerialization` (spațiul de nume `System.Xml.Serialization`) pentru serializarea XML.

Pentru a utiliza serializarea binară trebuie adăugat atributul `Serializable` în fiecare clasă care se doreşte a fi serializată. Atributul `Serializable` nu este moştenit, ceea ce înseamnă că, în cazul în care o clasă este marcată ca serializabilă, clasele derivate din acestea nu pot fi automat serializate. Acest lucru este logic deoarece aceste clase ar putea avea membri care nu sunt serializaţi. Pentru a marca faptul că un anumit cîmp nu este serializabil, se poate utiliza atributul `NonSerializable`. Spre deosebire de `Serializable`, atributul `NonSerializable` este însă moştenit. În procesul de serializare/deserializare binară de utilizează metodele `Serialize` şi `Deserialize` a obiectului `BinaryFormatter`.

În plus, dacă se doreşte definirea cîmpurilor de serializat precum şi controla modul de serializare în locul serializării predefinite, se poate implementa interfaţa `ISerializable`. Astfel, se poate defini o metodă de serializare proprie redefinind singurul membru al interfeţei `ISerializable`, şi anume metoda `GetObjectData`. Primul parametru al acestei metode, de tip `SerializationInfo` reprezintă o colecţie de tip nume/valoare care va fi transferată procesului de serializare; practic, va fi serializată doar informaţia adăugată în instanţa `SerializationInfo`.

Pentru deserializare, trebuie definit un constructor care primeşte aceiaşi parametri ca şi `GetObjectData` şi care va fi utilizat în procesul de deserializare, la crearea instanţelor obiectelor.

Serializarea unei instanţe a unei colecţii implică la rîndul său serializarea tuturor instanţelor elementelor din cadrul colecţiei. Această relaţie de dependenţă între clase reprezintă graful de obiecte al clasei respective. Mediul de execuţie .NET Framework parcurge graful de obiecte pentru fiecare clasă în procesul de serializare şi serializează toate obiectele acestuia. Este important în acest sens ca toate clasele componente din cadrul acestui graf să fie corect definite în vederea serializării.

Faţă de serializarea binară, în cadrul serializării XML sunt serializabile doar cîmpurile şi proprietăţile publice. Astfel, dacă datele instanţelor nu sunt accesibile din cîmpurile sau proprietăţile publice, ele nu vor fi iniţializate la deserializarea obiectelor.

Serializarea/deserializarea XML utilizează metodele `Serialize/Deserialize` a obiectului `XmlSerializer`.

Exercitiu:

Folosind noţiunile referitoare la serializare (XML şi binară) modificaţi aplicaţia de la laboratorul anterior astfel încât utilizatorul să poată serializa/deserializa un obiect `Serviciu` (idem şi pentru `Produs`). Adăugaţi capabilităţi de serializare/deserializare managerului `ProdusAbstractMgr`, după care modificaţi clasa `Program` pentru a permite utilizatorului să salveze/încarce lista de elemente (produse şi servicii) într-un fişier tip XML sau fişier binar.

Serializaţi/deserializaţi un obiect de tip Serviciu

1. Adăugaţi ***namespace***-urile următoare la clasa `Serviciu`:

```
using System.IO;
using System.Xml;
using System.Xml.Serialization;
```

2. Creaţi un constructor fără parametri (dacă nu există deja) în clasa `Serviciu`, `Produs` şi `ProdusAbstract` (aceşti constructori vor fi folosiţi de clasa `XmlSerializer`).

3. Adăugaţi şi metoda pentru serializarea XML , `save2XML`:

```
public void save2XML(string fileName)
{
    XmlSerializer xs = new XmlSerializer(typeof(Serviciu));
    StreamWriter sw = new StreamWriter(fileName + ".xml");
    xs.Serialize(sw, this);
    sw.Close();
 }
```

   - `XmlSerializer` este clasa care se ocupă de serializarea şi deserializarea obiectelor. Constructorul folosit primeşte un parametrul de tip `Type` – în cazul nostru, ştiind că obiectele sunt de tipul `Serviciu`, putem folosi :

```
typeof(Serviciu)
```

- `StreamWrite` este clasa care realizează scrierea datelor:

4. Modificaţi clasa `Program` pentru a salva primul serviciu din managerul de servicii.

5. Pentru că indexul managerului de produse şi servicii `ProduseAbstractMgr` returnează un obiect de tipul `ProdusAbstract` atunci când se doreşte accesul la un element din tabloul de elemente, iar metoda `save2XML` se află în clasa `Serviciu` trebuie să facem conversia la `Serviciu` – acest lucru se poate realizeaza cu ajutorul operatorului `as`:

```
(elemente[0] as Serviciu).save2XML(fileName);
```

6. Pentru a realiza deserializarea unui serviciu se poate utiliza metoda statică:

```
public static Serviciu loadFromXML(string fileName)
{
    XmlSerializer xs = new XmlSerializer(typeof(Serviciu));
    FileStream fs = new FileStream(fileName +".xml",FileMode.Open);
    XmlReader reader = new XmlTextReader(fs);

    Serviciu serviciu = (Serviciu)xs.Deserialize(reader);
    fs.Close();
    return serviciu;
}
```

7. Modificaţi clasa `Program` pentru a citi din fişier obiectul `Serviciu`:

```
Serviciu serv = Serviciu.loadFromXML(fileName);
Console.Write(serv.Descriere());
```

8. Inspectaţi conţinutul fisierul .xml în care aţi salvat obiectul `Serviciu`. Salvaţi fişierul sub un alt nume.

Particularizaţi salvarea fisierului XML

9. Adăugaţi namespace-urile folosite de serializare la clasa `ProdusAbstract`.
10. Adăugaţi atributul `XmlRoot` la clasa `Serviciu` astfel:

```
[XmlRoot("ServiciuParticularizat")]
public class Serviciu : ProdusAbstract,entitati.IPackageble
```

11. Adăugaţi atribute de serializare pentru fiecare câmp:

```
[XmlElement("ID")]
public long Id

[XmlElement("Numele")]
public String Nume

[XmlElement("CodulIntern")]
public String CodIntern
```

12. Inspectaţi fişierul rezultat în urma rulării programului. Comparaţi fişierul obţinut cu cel generat înainte de a folosi atributele de serializare.

Serializaţi/deserializaţi XML obiectele din managerul ProduseAbstractMgr

13. Adăugaţi ***namespace***-urile folosite pentru serializare la clasa `ProduseAbstractMgr`
14. Mutaţi funcţiile `save2XML` şi `loadFromXML` în clasa `ProduseAbstractMgr`
15. Modificaţi funcţiile `save2XML` şi `loadFromXML` pentru ca `XmlSerializer` să ştie că urmează să serializeze un tablou de elemente de tipul `Serviciu` şi `Produs`:

```
Type[] prodAbstractTypes = new Type[2];
prodAbstractTypes[0] = typeof(Serviciu);
prodAbstractTypes[1] = typeof(Produs);
XmlSerializer xs = new XmlSerializer
                     (typeof(ProdusAbstract[]),prodAbstractTypes);
```

16. Modificaţi funcţia `loadFromXML` pentru a
    - deserializa elementele nenule.
    - seta `nrElemente` = numărul de elemente nenule.

Serializaţi binar obiectele din ProduseAbstractMgr

17. Pentru a putea fi serializate binar obiectele de tipul `Produs` şi `Serviciu` trebuie să adăugăm atributul (serializare implicită):

    ```
    [Serializable]
    ```

    la declararea claselor.

18. Pentru a folosi clasa `BinaryFormatter` (clasă care formatează datele pentru a putea fi salvate) în clasa `ProduseAbstractMgr` trebuie să folosim:

    ```
    using System.Runtime.Serialization.Formatters.Binary;
    ```

19. Creaţi funcţia `save2File` care salvează elementele într-un fişier:

```
public void save2File(string fileName)
 {
  FileStream fs = new FileStream(fileName, FileMode.Create);
  BinaryFormatter formatter = new BinaryFormatter();
  formatter.Serialize(fs, this.elemente);
  fs.Close();
 }
```

20. Creaţi funcţia `loadFromFile` care încarcă în tablou elemente şi setează numărul de elemente.

Temă:

1. Creaţi în cadrul aplicaţiei implementate un meniu pentru a permite utilizatorului să salveze/încarce lista `elemente` într-un/dintr-un fişier XML specificat, cu tratarea corespunzătoare a excepţiilor (de exemplu, imposibilitate de citire sau scriere din/în fişier). Afişaţi conţinutul fişierului XML.

2. Incercați sa serializați XML un obiect de tipul `List` – de exemplu (`List<ProdusAbstract>` sau `List<IPackageble>`) şi utilizați această serializare pentru a serializa un obiect `Pachet` şi/sau întreaga mulțime de obiecte `Pachet` din cadrul aplicației dezvoltate în cadrul Laboratorului nr. 6 (în cadrul căruia au fost utilizată structura `List` pentru memorarea elementelor dintr-un pachet şi al pachetelor). Inspectați fişierul generat în procesul de serializare.